

Linac Front-end Software

Comparison between MC68040 and PowerPC

Robert Goodwin, Robert Peters

Tue, Jul 30, 2002

The front-end software, written for the Motorola 68K CPU family and used for many years for control system support in Linac and other Fermilab projects, has been converted to run on the PowerPC. During the conversion, a strong effort was made to preserve the same features that are supported in the original implementation. As a result, the list of supported features is nearly identical, so that either may be used for a given project. This note is an attempt to compare and contrast the two implementations.

Hardware implementations

The MC68040-based nodes are usually delivered in the Internet Rack Monitor (IRM) configuration, in which a rack-mounted 3U crate is outfitted with I/O connectors, ethernet, Tevatron clock, and houses the MVME-162 CPU board in a 3-slot VME crate, which allows for some expansion. One required component of such a configuration is a Digital IP board, which uses one of the four available IndustryPack (IP) slots of the CPU board, which uses a 25 MHz clock and includes 4 MB of dynamic RAM and 0.5 MB of nonvolatile memory. The Digital IP board, along with an associated digital interface board mounted within the crate configuration, provides support for 8 bytes of digital I/O and also clock decoding that results in a cycle interrupt that drives the periodic accelerator-synchronous activities of the software. (It also provides interrupts for all clock events.) To support analog I/O, another IP board is used in connection with an analog interface board. This board supports 64 A/D channels, all of which are automatically digitized at 1 KHz and placed into a circular buffer (large enough to contain 512 sets of data), plus 8 D/A channels. The two additional IP slots that are available may be used for timing channels or high-speed digitizers. The two spare VME slots are seldom used.

The PowerPC implementation is based on the MVME-2401 CPU board that uses a 233 MHz clock, 32 MB of dynamic RAM, and two PMC slots. In all configurations, both slots are occupied by a 2 MB nonvolatile RAM board and a Digital PMC board, which functions much like the Digital IP board above, but it also includes 8 timers. To support analog I/O, one can use an IP carrier board in a VME slot to hold the necessary IP board. In the current implementation used in Linac, the analog I/O is interfaced through Smart Rack Monitors (SRMs) that communicate via arcnet with the VME crate housing the CPU board.

The two implementations use CPUs of very different clock rates. But the PowerPC boards have relatively slow access times ($\sim 1 \mu\text{s}$) to both the nonvolatile memory and the Digital PMC board. This means that heavy access to nonvolatile memory can result in a large cost in PowerPC performance. In order to improve this situation, and thereby regain some loss of performance, a number of changes have been made to the system code. Searches of nonvolatile memory tables have been optimized, and algorithms have been rewritten to minimize the number of slow accesses needed. These changes have been made to both implementations. The resulting performance measurements are described later.

Major common elements

In nearly all respects, operation and software organization of both implementations are identical, even though the IRM system software is written in assembly code and the PowerPC system software is written in C. This is no accident at all; rather, it was one of the key strategies used during the PowerPC conversion effort. Both systems support the Classic

protocols and the Acnet suite of protocols that are used in the Fermilab accelerator control system. The host level Acnet nodes, such as the accelerator console displays, use the Acnet protocols, including ACNAUX, RETDAT, SETDAT, FTPMAN, and ALARMR. The low level diagnostic programs that utilize the Classic protocols—supporting data requests, data settings, and alarm reporting—are used by programs that run on various and sundry client platforms, including Macintosh, PC, Unix, and the “little consoles” supported by direct hardware connection between a front-end and console display hardware, or indirectly by software emulation of such displays on the same set of client platforms. These Classic protocol clients are used as a low level method of access to these front-ends to assure consistent and reliable operation and to assist with problem diagnosis. Both sets of protocols are based upon UDP/IP to insure timely responses in a 15 Hz accelerator environment. UDP multicasting is used heavily by these front ends to distribute data requests or alarm messages spanning multiple nodes, either to all front-ends in an entire project, or in the case of the time-of-day, to the front-ends in all projects.

At Fermilab, full support of the Acnet protocols includes support for the Tevatron clock event system, which delivers up to 256 synchronizing signals encoded on a 10 MHz carrier to all parts of the accelerator complex. It is this clock event system that allows all front-ends in a project to operate in synchronism with each other at 15 Hz. It is this synchronous operation that makes possible delivery of correlated data across multiple front-ends to clients, which is a fundamental goal of the original system design. To assist clients that cannot operate in real-time synchronism with the accelerator clock system, support for server-style requests is provided for both Classic and Acnet protocols, in which a request for a set of devices spanning multiple target nodes is sent by a client to any front-end, which in turn acts as a server to forward the request to all target nodes using multicasting, receive replies from all contributing nodes, and deliver a composite reply to the original client. When the client receives the reply, the data included is correlated across all target nodes, meaning that it was measured on the same 15 Hz accelerator cycle. Requests specifying periodic replies result in an immediate first reply, followed by periodic replies that are synchronized with the normal cyclic operation of the front-ends. One-shot requests merely result in the immediate reply. (Requests that specify returns on a clock event result in no immediate response, but only replies occurring on cycles after detection of the specified clock event.)

Configuration of these front-ends includes installing appropriate data in nonvolatile memory tables, so that when a front-end is reset, perhaps following a power outage, it knows all it needs to know to again take on its unique characterization and personality as a component of the accelerator control system. When a node is reset, it obtains a copy of the system code from a common server via TFTP, but its knowledge of the specific devices to which it is connected comes from information retained in its nonvolatile memory system tables. Local application programs are also stored in this nonvolatile memory, achieving rapid and automatic activation of those programs that act as an extension of the system code to further characterize that node's operation. In summary, most of the software is common across all front-ends, but the particular suite of local applications installed may vary.

The cyclical pattern of operation is common across both front-end implementations and is supported by three tasks. A cycle interrupt, generally occurring at 10 Hz or 15 Hz, causes the Update task to run to refresh the local data pool of analog and digital data by accessing its own I/O interfaces and performing any closed loop algorithms that may be supported by local applications, after which it builds reply messages for all active data requests for which a reply is due on that cycle, queuing each such reply message to the network. Another task

then performs the alarm scan, checking the data pool readings for all devices for which alarm scanning is enabled, after which it queues any resulting alarm messages to the network using the Classic protocol. (Acnet protocol alarms are delivered by a special local application that serves to shepherd them to the Acnet alarm handler.) The third task then runs to invoke the single active page application so it can perform its duties for that cycle. (The page application supports a “little console” user interface that may be hardware and/or software-emulated.)

MC68040-based nodes

The execution time for these three tasks can vary widely. In a typical IRM, the Update task completes in less than 5 ms. The Alarms task uses about $(120 + 5*N)$ μ s, where N is the number of devices enabled for alarm scanning. For 50 devices, this is about 0.3 ms. Most IRMs have much fewer devices enabled for alarm scanning.

Of course, the execution time of a page application depends upon what it is has to do, but as an example, the parameter page application uses 0.4 ms on most cycles, with 1.6 ms needed when it updates its display at about 1 Hz. In the case that it is displaying data from other nodes, it can take longer, as it tries to be patient about receiving such external data each cycle. One example of that is an IRM parameter page display showing data from 9 different Linac PowerPC nodes, which might take about 13 ms due to this patience. The reason for such lengthy times required to collect data from the speedy PowerPC nodes is described in the next paragraph; however, such patience yields a display that always shows correlated data across all nodes. In the absence of the need for such page application patience, a typical IRM is busy less than 10% of the time. The spare time allows support for unusually heavy request activity when needed, all the while continuing to support 15 Hz accelerator activity.

PowerPC-based nodes

The Linac PowerPC nodes, in executing these same three tasks, spend a much longer time in the Update task, because most of the data comes from lower level front-ends called Smart Rack Monitors, or SRMs. At the beginning of each cycle, the front-end sends a request message to each SRM that causes it to read all of its hardware I/O interfaces and build a data pool that it then returns (over arcnet) to the front-end in response to its request. During the time that the SRMs are busy, the PowerPC front-end can only await the results. It cannot continue its cyclical jobs until it has the latest raw data from the SRMs, since some of those jobs need to use the latest data. It is this waiting logic in the Update task that ensures that an external requester cannot sample data from a partially-updated data pool. To an external client, the front-end's data pool is updated all at once on each 15 Hz cycle. The time spent waiting for the SRM replies is typically 10–15 ms. The number of SRMs ranges from 0–6.

The Alarms task typically executes in less than 0.7 ms, for a case of 150 devices checked. The time is 1.1 ms for 240 devices. The formula might be about $(40 + 4*N)$ μ s. The parameters needed for performing the alarm scan—reading, nominal, tolerance, flags—are stored in nonvolatile memory, so this speed is not dramatically better than that of the IRM. But scanning alarms for 200-odd devices in 1 ms is hardly awful.

For the page application task, using the same example of the parameter page application displaying a list of local parameters, the time is less than 0.1 ms, with 0.7 ms needed when the display is updated. For the case of displaying data from another node, the typical time is still less than 1 ms, because both nodes had already spent time earlier in that cycle waiting for replies from their connected SRMs.